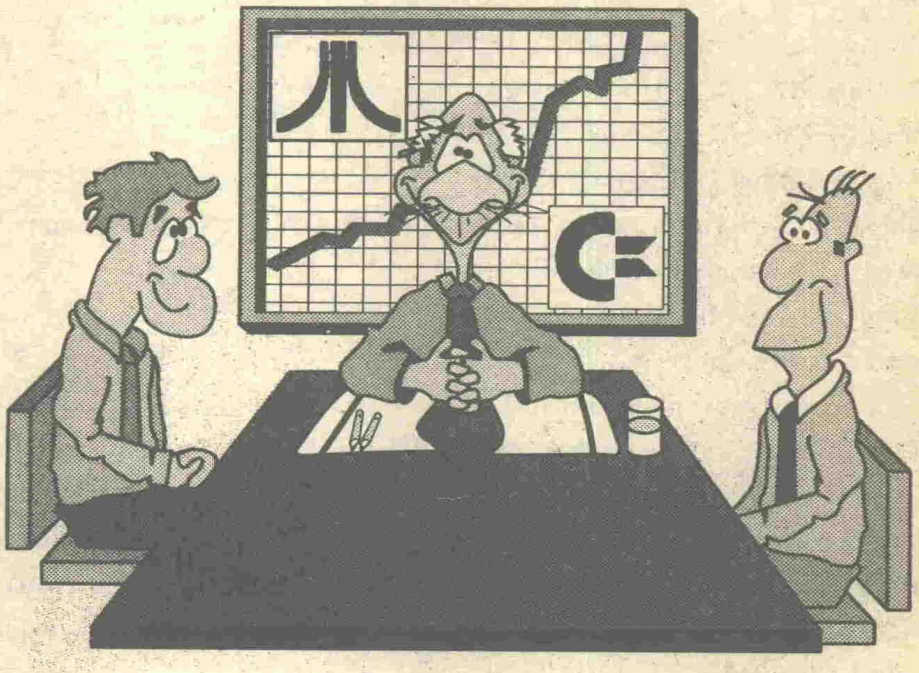
ปัญหาชวนตื้อ สำหรับผู้เริ่มเล่น Visual Basic ตอนที่ 2

ชี้ให้เห็นถึง หลักการโค้ดของ มืออาชีพที่เขาคำนึงถึง ก่อนลงมือเขียนโปรแกรม



ข้อดีที่สุดของ VB ก็เห็นจะได้แก่เรื่องของความง่ายในการสร้างแอพพลิเคชันให้เสร็จออกมาทันอกทันใจ ถึงแม้ผู้เขียนโปรแกรมจะมีความเชี่ยวชาญอยู่น้อย ประสบการณ์เขียนโค้ดไม่ถึงปี แล้วก็ความรู้ในเรื่องลึกต่ำก็เป็นไปได้ที่จะเขียนโปรแกรมเล็ก ๆ ที่เป็นประโยชน์ออกมาสักชิ้นภายในวันสองวัน

จริง ๆ แล้ว VB ยอมรับโปรแกรมเมอร์ทุกประเภท ทั้งพวกเขียนโปรแกรมอย่างมีโครงสร้าง มีกำหนดลำดับแน่นอน และทั้งผู้นิยมชมชอบสปาร์เกตตี้ หมายถึงนึกอะไรได้ก็จับใส่เพิ่มเข้าไปเรื่อย ๆ จนกระทั่งโค้ดมันพันกันยุ่งอีลุงตุงนังเหมือนเส้นสปาร์เกตตี้ในจานอาหารกลางวัน อย่างที่ฝรั่งเขาเรียกสปาร์เกตตี้โค้ด (ถ้าคนไทยพูดก็น่าจะเรียกการเขียนสไตล์นี้ว่าโค้ดเส้นหมี่มากกว่านะครับ)

ที่ VB สามารถรับได้ทั้งสองประเภทก็เนื่องจากโครงสร้างชุดโปรแกรมบังคับอยู่ในตัวให้เขียนโค้ดในลักษณะของโมดูล เขียนอย่างรู้ดีว่ากำลังสั่งออปเจ็กต์ตัวใดให้ทำหน้าที่อะไรในเวลาไหน

มือโปรแกรมสมัครเล่นจึงสืบหาที่อยู่ของตัวปัญหาของตนได้ไม่ยากนัก และโปรแกรมเมอร์มือฉมังก็สามารถวางแผนโค้ดที่สลับซับซ้อนได้อย่างเป็นระบบระเบียบ เขียนไปเป็นปีกลับมาอ่านคอมเมนต์ของตัวเองหน่อยเดียวก็จำได้ทะลุปรุโปร่งว่าอะไรเป็นอะไร อยู่ที่ไหน จะต้องแก้ไขประการใดจึงจะเมนเทนให้เข้าสู่สภาพใหม่ที่ต้องการ

ในตอนนี้จึงอยากจะชี้ให้เห็นหลักการโค้ดที่มืออาชีพเขาคำนึงถึงกันก่อนลงมือเขียน อาจไม่ครบถ้วนทุกกระบวนความเทคนิคทั้งหมดหรอกนะครับ แต่อย่างน้อยก็คิดว่าคงพอจะเป็นแนวทางสำหรับผู้เริ่มต้นได้บ้าง

ชื่อนั้นสำคัญไฉน?

จริง ๆ แล้วปัญหานี้ได้รับการแนะนำไปแล้วในตอนต้น ๆ ของโปรแกรมเมอร์ไกด์ แต่ไม่ค่อยมีใครสนใจ โดยเฉพาะคนที่เพิ่งเข้ามาสู่วงการโปรแกรมแล้วก็เริ่มจับ VB เลยเป็นตัวแรก ส่วนใหญ่จะนำเอาออปเจ็กต์ที่ต้องการไปใส่ไว้ในฟอร์ม ตั้งค่า properties แล้วก็ตั้งหน้าตั้งตาเขียนโค้ดกันอย่างขะมักเขม้น เนื่องจากออปเจ็กต์แต่ละ

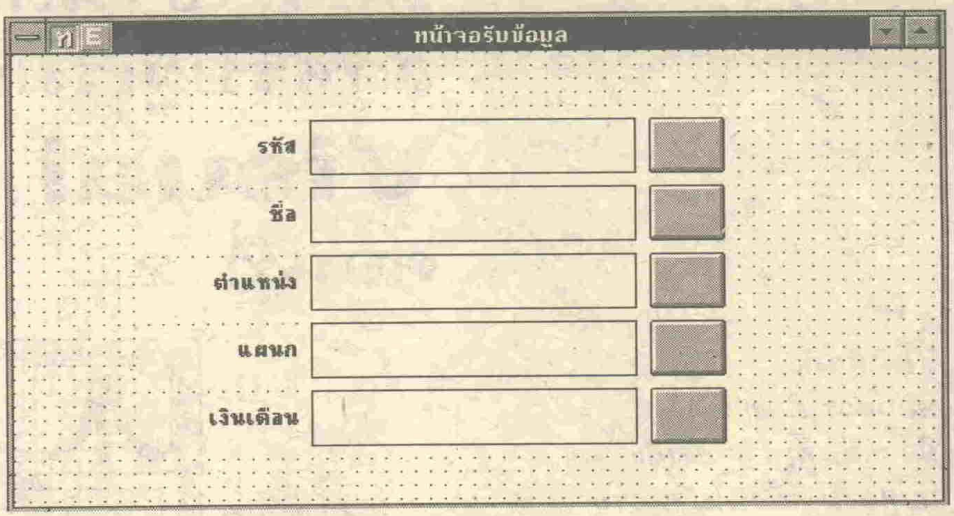
ตัวมีชื่อดีฟอลต์อยู่แล้ว เพียงอ้างชื่อที่เขาให้มาในโค้ดก็เห็นว่าใช้การได้ ไม่มีปัญหาอะไร ไม่เกิดบักหรือความขัดแย้งใด ๆ ทั้งสิ้น

นั่นก็ถูกอยู่หรอกครับ และก็น่าสนับสนุนด้วยสำหรับโปรแกรมประเภทมีปุ่มสองปุ่ม มีเท็กซ์บ็อกซ์กล่องเดียว เพราะถ้าไม่เมาคุณต้องจำได้แน่นอนว่ามันชื่อ Text1, Command1 หรือ Command2 คุณจะไม่ต้องปวดหัวกับการไล่ชื่อแซ่ของพวกมันที่เรียงลำดับอักษรอยู่ในกล่อง Object และคุณก็จะไม่ต้องกังวลว่าจะสับสนเรียกใช้พวกมันผิด ๆ ในโค้ด ทุกอย่างมันง่ายและเห็น ๆ กันอยู่ เร่ง ๆ ทำให้เสร็จ ๆ โดยไม่ต้องเสียเวลามาตั้งชื่อดีกว่า

แต่หากในฟอร์มเดียวคุณต้องมีปุ่มสักสิบปุ่ม มีเท็กซ์บ็อกซ์ซะยี่สิบ แถมออปชันอีกสามแผงสลับกันปรากฏตัวหยอดอีกหน่อยด้วยไทม์เมอร์ซึ่งทำหน้าที่ต่างกันอีกห้าหน่วย อย่างนี้ใครจำได้ว่าออปเจ็กต์ตัวไหนทำหน้าที่อะไรก็เทวดาแล้วล่ะครับ โดยเฉพาะพักไปเสียสามเดือนกลับมาดูใหม่ ร้อยทั้งร้อยต้องมานั่งเรียกฟอร์มเพื่อขอดูชื่อออปเจ็กต์ระหว่างโค้ดไปด้วย

มันมีความจริงสัมพันธ์อยู่อย่างว่าแอพพลิเคชันไหนมีออปเจ็กต์เยอะ โค้ดที่ใช้ก็จะเยอะตามไปด้วย ไม่ว่าจะเยอะที่ตัวออปเจ็กต์เอง หรือเยอะที่โพรซีเดอร์ เราจึงน่าจะมีวิธีการตั้งชื่อให้เกิดความระลึกได้ทันทีว่ามันคือออปเจ็กต์ประเภทไหน ทำหน้าที่อะไร และมีความสัมพันธ์อย่างไรกับออปเจ็กต์ตัวอื่น ๆ แค่เห็นชื่อเท่านั้นนะครับนี่!

หากเป็นภาษาคอมพิวเตอร์สมัยเจ้าคุณปู่ก็หมดทางล่ะครับ แต่สมัยของคุณมันเป็นไปได้จริง เพียงแต่คุณรู้หลักการที่ว่า



รูปที่ 1 แสดงหน้าจอ input ที่ออกแบบขึ้น

● ออปเจ็กต์แต่ละประเภทมีชื่อเรียกของตัวเอง สามารถนำมาย่อให้เหลือแค่อักษรสามตัวได้ คุณเองหรือใครมาเห็นก็ระลึกได้ว่ามันคือออปเจ็กต์ชนิดใด เป็นต้นว่าเห็นแค่ cmd เท่านี้ก็พอเดาได้เป็นเค้าแล้วว่ามันคือ Command Button

● เหตุการณ์ที่เกิดขึ้นในโค้ดสามารถใช้ภาษาอังกฤษมาสื่อได้ และคำศัพท์หลายคำที่ใช้สื่อก็อาจย่อให้เหลือพอเป็นเค้าเช่นเดียวกัน ไม่จำเป็นต้องเอามาทั้งดุ้น เป็นต้นว่าต้องการใส่ข้อมูลรายจ่ายสะสม เพียงรู้จักภาษาหน่อยเดียวคุณก็นึกได้ทันทีว่าควรเป็น Accumulated Expense ทว่ายาวแบบนี้เห็นทีจะไม่เหมาะกับการใช้อ้างในโค้ด ก็ตัดให้สั้นพอเหลือสัณฐานเป็น AccExp เสีย อย่างนี้อ่านเท่าไหร่ก็ยังรู้สึกว่ามันคือ Accumulated Expense อยู่ดีนั่นแหละ

● คุณสามารถนำเหตุการณ์ดังกล่าวมาตั้งชื่อให้ออปเจ็กต์ได้มากกว่าหนึ่งตัว แถมเอาไปใช้ซ้ำในโพรซีเดอร์ก็ยังได้ ด้วยการนำชื่อย่อของประเภทออปเจ็กต์มานำหน้าชื่อเหตุการณ์ พูดง่าย ๆ ว่าใช้ prefix เสียหน่อยเดียวเท่านั้น คุณก็ใช้ชื่อเหตุการณ์ได้ไม่รู้จบ

ผมจะยกตัวอย่างที่ทุกคนคุ้นเคยกันมาตั้งแต่สมัยเรียน และก็ติดเป็นเงาตามตัวมาถึงวัยทำงานด้วย นั่นคือการใส่ข้อมูลเข้าเทเบิลของพนักงาน ซึ่งเป็นปกติที่ต้องมีกลุ่มฟิลด์ของแต่ละเรคอร์ดเป็นพวก รหัส ชื่อนามสกุล แผนก ตำแหน่ง แล้วก็เงินดาวน์เงินเดือนอะไรทำนองนั้น

ผมจะสมมติว่าคุณออกแบบหน้าจอ input ให้มีลักษณะดังในรูปที่ 1

คุณกำหนดให้มีการกดปุ่ม (อันนี้อาจใช้เทคนิคได้นิดหน่อย คือเช็คดูว่าเมื่อเกิดการเคาะ Enter ที่เท็กซ์บ็อกซ์ตัวใดก็ให้เรียกใช้โพรซีเดอร์ Command_Click ในกรณีที่ผู้ใช้อาจสะดวกที่จะใช้คีย์บอร์ดมากกว่า) แล้วเกิดการโฟกัสไปที่เท็กซ์บ็อกซ์ซึ่งทำหน้าที่รับข้อมูลของแต่ละฟิลด์ แถมยังมีการไฮไลต์ข้อมูลเก่าเพื่อรอข้อมูลใหม่มา

แทนที่ ถ้ารวมเลเบลเข้าไปด้วยก็จะเห็นว่าออปเจ็กต์ในนี้มีอยู่ทั้งหมดสิบห้าตัว (ไม่รวมฟอร์ม) แต่จะดีกว่าไหมถ้าหากคุณจะรู้สึกว่ามันมีอยู่แค่ห้า

สำหรับผู้ที่ชื่นชอบรสชาดการแก้โค้ดเป็นวรรคเป็นเวรคงให้คำตอบได้ว่าย่อมต้องดีกว่ามาก เพราะจะได้ไม่ต้องจดจำชื่อเฉพาะตัวของแต่ละออปเจ็กต์ให้มากความ

ขออนุญาตกล่าวย้อนไปเปรียบเทียบกับวิธีการแบบเก่าสักหน่อย หากคุณทอดหุ่ย ปล่อยให้ VB ทำการตั้งชื่อออปเจ็กต์ให้โดยอัตโนมัติ คุณจะเห็นอะไรครับเมื่อกดปุ่ม Object List ที่ข้างกล่อง Object แน่นอน ต้องได้เห็นอย่างในรูปที่ 2

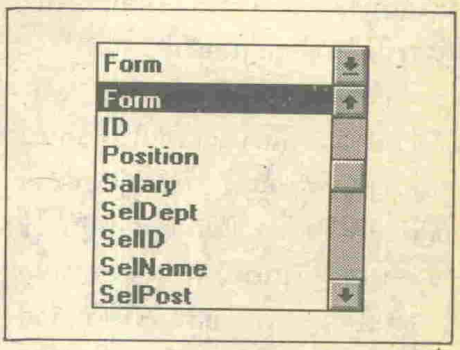
จริง ๆ แล้วยาวลงไปข้างล่างมากกว่านี้อีก ขนาดคุณยังไม่ได้เห็นกลุ่ม Text เลยนะนี่ ดูไม่รู้เรื่องเท่าไหร่ใช่ไหมครับว่าอะไรเป็นอะไร คุณต้องมานั่งเทียบด้วยสายตาตัวต่อตัวว่า Command1 หมายถึงฟิลด์อะไร มีความสัมพันธ์กับ Label1 หรือไม่? ยิ่งถ้าคุณปล่อยโค้ดตัวนี้ไปแล้วหลายเดือน พอย้อนกลับมาดูอีกทีจะรู้สึกว่างเปล่าเป็นอย่างยิ่ง งงครับ อะไรคืออะไรอยู่ที่ไหน ต้องเหนื่อยไล่กันน่าดูชม

และอีกทางหนึ่ง สมมติว่าคุณตั้งชื่อให้ออปเจ็กต์แต่ละตัวแล้วล่ะ ทว่าเป็นอย่างที่เห็นในรูปที่ 3 คุณก็ยังไม่หายงงอยู่ดี อย่างน้อยก็ต้องมานั่งระลึกสักแว่บหนึ่งว่าชื่ออย่างนี้หมายถึงออปเจ็กต์ชนิดใด

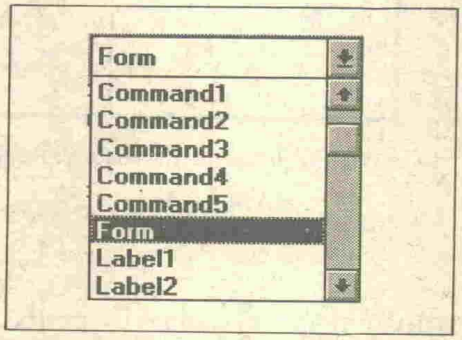
แต่หากลองตั้งชื่อตามในตารางที่ 1 ดูล่ะครับ อะไรจะเกิดขึ้น

ถ้าเอาตามในตารางที่ 1 จะเห็นได้ว่าเหมือนคุณมีออปเจ็กต์อยู่แค่ห้าตัวเท่านั้น คือออปเจ็กต์ที่เกี่ยวกับรหัส (ID) ชื่อพนักงาน (Name) ตำแหน่ง (Post) แผนก (Dept) และเงินเดือน (Sal) ส่วนจะเป็นชนิดไหนก็สามารถดูได้ที่ prefix หน้าชื่อ และจากการใช้ภาษามาสื่อตรงตามชื่อฟิลด์ที่ต้องการ คุณก็ไม่ต้องไปค้นหาให้เสียเวลาว่าชื่อออปเจ็กต์นั้น ๆ คืออะไร อยู่ที่ไหน และมีความเกี่ยวข้องกับเทเบิลข้อมูลพนักงานอย่างไร

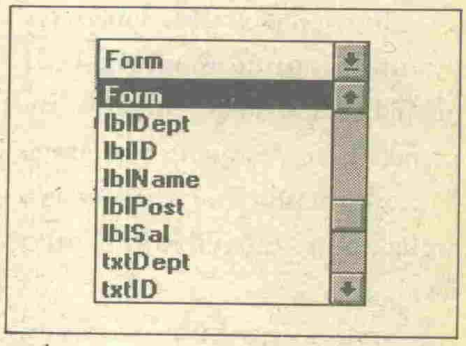
แล้วคราวนี้เมื่อคุณกดปุ่มเรียกดูรายชื่อออปเจ็กต์ที่กล่อง Object คุณก็



รูปที่ 3



รูปที่ 2 แสดงหน้าจอเมื่อคุณกดปุ่ม Object List ที่ข้างกล่อง Object



รูปที่ 4

Objects	Responding Fields	Name
Text Box	รหัส	txtID
Text Box	ชื่อ	txtName
Text Box	ตำแหน่ง	txtPost
Text Box	แผนก	txtDept
Text Box	เงินเดือน	txSal
Command Button	รหัส	cmdID
Command Button	ชื่อ	cmdName
Command Button	ตำแหน่ง	cmdPost
Command Button	แผนก	cmdDept
Command Button	เงินเดือน	cmdSal
Label	รหัส	lblID
Label	ชื่อ	lblName
Label	ตำแหน่ง	lblPost
Label	แผนก	lblDept
Label	เงินเดือน	lblSal

ตารางที่ 1

จะได้สบายตาสบายใจ ไม่ต้องมานั่งเล็งระลึกให้เมื่อยอีกต่อไป

ด้วยความที่มีลักษณะเป็น self-doccmented นี้เอง คุณจึงเห็นพวกมันรวมกลุ่มกันเป็นอันดี ป้ายก็อยู่ส่วนป้าย ช่องก็อยู่ส่วนช่อง ปุ่มก็อยู่ส่วนปุ่ม ใช้ครั้งแรกอาจสับสนนิดหน่อยตรงที่ชื่อมันดูเหมือน ๆ กัน แต่พอชินตาแล้วก็จะรู้ว่าควรไปหาที่ไหน ก็ prefix มันบอกอยู่นี่ครับ

ในกรณีที่คุณจะมีโพรซีเดอร์อิสระประเภทที่ทำงานให้กับฟิลด์ใดฟิลด์หนึ่ง คุณก็ยังสามารถใช้ชื่อเก่าได้อยู่อีก ง่ายนิดเดียว ไม่อยากจะพูดเลย รู้ว่าพวกคุณถึงบางอ้อกันไปนานแล้ว ก็แค่ตัดชื่อ prefix ออก เหลือแค่ชื่อฟิลด์ด้วน ๆ ก็พอ

อาจเกิดปัญหาขึ้นมาในกรณีชื่อที่คุณปรารถนาใช้ตั้งชื่อโพรซีเดอร์แต่มันเกิดไปเป็นคำสงวนเข้า เป็นต้นว่า Date (เพื่อแสดงว่าพนักงานเริ่มทำงานเมื่อไหร่) อย่างนี้ เราจะแก้ปัญหาได้อย่างไร?

มีความเป็นไปได้สองทางครับ คือ คุณต้องตั้งชื่อ prefix ขึ้นมานำหน้าชื่อ เช่น prdDate ซึ่งหมายถึงว่า Date ตัวนี้มันเป็นโพรซีเดอร์นะ ไม่ใช่ฟังก์ชันเรียกเวลา หรือให้เข้าทีกว่านั้นหน่อยก็ตกแต่งชื่อเสียงใหม่เป็น BegDate ตรงความหมายเดิมด้วย และก็ไม่ซ้ำซ้อนกับคำสงวนให้เสียอารมณ์ด้วย มันดีอย่างนี้ล่ะครับ

ผมขอยกเอา prefix ของออปเจ็กต์ทั้งหมดมากล่าวไว้ในที่นี้เลย (ในตารางที่ 2) คุณจะได้ไม่ต้องไปเปิดหาในเท็กซ์ที่ไหนอีก

ต่อไปพอจะใช้จริง ๆ อาจเกิดความขี้เกียจหน่อยในระยะแรกนะครับ แต่พอชินแล้วและเห็นประโยชน์อย่างมากมายของการตั้งชื่ออย่างมี prefix คุณก็จะรับได้ว่ายอมออกแรงอีกหน่อยแต่ได้โค้ดที่เป็นระเบียบ อ่านง่าย เมนเทนง่ายนั้นมันแสนคุ้ม เนื่องจากในความเป็นจริงเราต้องมีออปเจ็กต์มากกว่าที่ยกตัวอย่างหลายกระบุง เป็นต้นว่าปุ่มจบการทำงานอย่างนี้ เมนูเลือกฟอร์มหลักฟอร์มย่อยอย่างนี้ หรือจำนวนฟิลด์ต่อหนึ่งเรคอร์ดมากกว่าสิบอย่างนี้ ขืนคุณปล่อยให้ VB มันตั้งชื่อ หรือคุณตั้งชื่อแบบไม่มี prefix คุณก็เตรียมตัวเสียเวลาไล่ชื่อด้วยลูกกะตาได้เลยครับสนุกแน่

ส่วนใครที่ขยันพิมพ์มาก ๆ ไม่กลัวชื่อยาว กลัวแต่ความจำกัดอักษรก็สบายใจได้นะครับ คุณพิมพ์ไปเถอะทางไมโครซอฟต์อนุญาตให้คุณขยันคีย์ได้ถึง 40 ตัว ใครบ้าจี้พิมพ์ไปเต็มอัตราที่เขากำหนดไว้ล่ะก็บอกมาด้วยนะครับ ผมจะไปคุกเข่าคำนับด้วยความเคารพหามานานแล้วคนตั้งชื่อตัวแปรหรือออปเจ็กต์ยาว 40 ตัวอักษร

Object	Prefix	Object	Prefix
Form	frm	Label	lbl
Check box	chk	Line	lin
Command Button	cmd	List box	lst
Directory list box	dir	Menu	mnu
Drive list box	drv	Option button	opt
File list box	fil	OLE client	ole
Frame	fra	Picture box	pic
Grid	grd	Shape	shp
Horizontal scroll bar	hsb	Text box	txt
Vertical scroll bar	vsb	Timer	tmr
Image	img		

ตารางที่ 2 แสดง prefix ของออปเจ็กต์ทั้งหมด

สิ่งที่ควรเห็นใน Form_Load คืออะไร?

เป็นอีกเทคนิคเล็ก ๆ น้อย ๆ เพื่อความมีโค้ดที่อ่านง่ายและเมนเทนง่ายนะครับ ลองคิดดูว่าการนั่งไล่โค้ดที่ลืมไปแล้วของตัวเองก็ดี หรือการต้องไปนั่งไล่โค้ดของชาวบ้านที่เขายัดเยียดมาให้คุณแก้ก็ดี ทุกคนต้องมาเริ่มที่ Form_Load กันทั้งนั้น ด้วยเหตุที่มันเป็นจุดเริ่มต้นของทุกสิ่งทุกอย่าง ถ้าคุณต้องการสืบสาวโค้ดจากต้นไปหาปลายทะลุปรุโปร่ง คุณก็ต้องมาเริ่มกันที่นี่แหละ จากนั้นถึงจะสาวไปหาขั้นตอนการทำงานอื่น ๆ ดูไปเรื่อย ๆ ว่าออปเจ็กต์ตัวไหนมี properties อย่างไร ข้างในเขาใส่โค้ดอะไรไว้บ้าง (ซึ่งตรงนี้อาจต้องดูละเอียดหน่อยนะครับ บางทีมีการหมกเม็ดไว้ คือออปเจ็กต์ตัวหนึ่งมีโค้ดอยู่ในหลายเหตุการณ์ ต้องคลิ๊กดูในช่องเหตุการณ์ให้ละเอียด)

โปรแกรมจะตื้นลึกหนาบางอย่างไร

ก็ต้องเข้าประตูแรกคือ Form_Load นี่จึงเป็นประเด็นที่ผมอยากนำขึ้นมาถกว่าเราควรเห็นอะไรที่นี่บ้าง

หลายคนหลายคราวใส่โค้ดหลายบรรทัดลงไปในนี้ ซึ่งก็มีหลายทีหลายเวลาที่กลับมาดูโค้ดตัวเองแล้วมึนอยู่เป็นนาน เพราะเห็นอะไรไม่รู้จับฉ่ายไปหมด ธรรมดาแอพพลิเคชันใหญ่เมื่อโหลดขึ้นมาทีแรกก็มักจะจัดการงานหลายชิ้น ติดต่อสั่งเสียกับออปเจ็กต์หลายตัวอยู่แล้ว ก็แน่ล่ะครับที่คุณจะได้เห็นอะไรต่ออะไรซ้อนกันมั่วไปหมด ถ้านำไปใส่ไว้ในโพรซีเดอร์ฟอร์มโหลดที่เดียว

จึงมาถึงคำแนะนำที่ว่า ถ้ามีงานหนึ่งต้องใช้โค้ดเกินหนึ่งบรรทัดแล้วล่ะก็ เก็บมันเป็นโพรซีเดอร์อิสระเถอะครับ ตั้งชื่อให้เกิดความระลึกได้ว่าคุณเรียกใช้มันแล้วจะให้มันทำอะไรบ้าง ต่อไปคุณเกิดข้อสงสัยว่าฟอร์มนั้นโหลดแล้วทำอะไรบ้าง ก็เป็นอันว่าได้อ่านกันอย่างสบาย ผมจะยกตัวอย่างให้เห็นสักนิดอย่างนี้

```
Sub Form_Load ()
   Me.WindowState = 2        _____(1)
   SetGrid                   _____(2)
   FetData                   _____(3)
End Sub
```

ในบรรทัดที่ (1) นั้น จะเห็นได้ว่าเป็นการจัดเรื่องเกี่ยวกับหน้าต่างเรื่องเดียว คือให้มันขยายออกเต็มขนาดอย่างที่เรียกแม็กซิไมซ์นั่นแหละครับ อันนี้เราเขียนลงไปตรง ๆ ไม่ต้องไปเสียเวลาเรียกโพรซีเดอร์ให้ลำบาก เพราะเห็นตำตาว่ามันเป็นเรื่องเกี่ยวกับอะไร

ส่วนบรรทัดที่ (2) นั้น ถ้าคุณเคยยุ่งเกี่ยวกับกริดมาบ้าง คงทราบดีว่ามันกินที่หลายบรรทัดกว่าจะทำให้มันมีจำนวนแถวแนว ขนาดสูงกว้าง ขนาดของแต่ละแถวแนว จะใช้ฟอนต์อะไร

ให้มีเส้นหรือไม่ เอาแถบมืดหรือเปล่า รวมทั้งสิ่งสำคัญอีกอย่างที่ขาดไม่ได้คือหัวข้อรายการของแต่ละแถวแนว จึงควรใช่ไหมครับที่คุณจะทำให้มันกลายเป็นโพรซีเดอร์หนึ่งเสียเลย

และมาถึงบรรทัดสุดท้ายคือ (3) อันนี้ยิ่งยุ่งครับ คงเดาออกกันถ้วนหน้านะครับว่า FetData นั้นย่อและรวมศัพท์สองคำไว้ด้วยกันคือ Fetch กับ Data ซึ่งหมายถึงการดึงข้อมูลขึ้นมานั่นเอง ถ้าหากเป็นการดึงข้อมูลจากหลายเทเบิลก็จะมีหัวเลี้ยวหัวต่ออยู่ในนี้มากมาย อาจเป็นโพรซีเดอร์ที่เรียกโพรซีเดอร์อิสระอื่นมาใช้ หรือสั่งให้ออปเจ็กต์หลายตัวทำงาน ขึ้นอยู่กับ requirement ของผู้ใช้ ยิ่งถ้าเกี่ยวข้องกับกราฟล่ะก็ยิ่งแล้วใหญ่เลย ขืนคุณเอาสเตทเมนต์เหล่านั้นทั้งหมดมาสุมไว้รวม ๆ กันในหน้าจอ Form_Load อย่างเดียว มีหวังวุ่นน่าดูชม โผล่ขึ้นมาก็อ่านกันตาเหลือกแล้ว

บางคนอาจเห็นว่ามันมีสามบรรทัดยังเยอะไปอยู่ดีด้วยซ้ำ อยากตัดให้เหลือแค่สองบรรทัดพอ เพราะคิดว่าไหน ๆ สองบรรทัดแรกก็ถูกเรียกแค่ครั้งเดียวอยู่แล้ว อย่ากระนั้นเลย รวบหัวรวบท้ายมาไว้ด้วยกันทีเดียวจะดีกว่า

ก็ได้ครับ เราคิดชื่อเสียใหม่ให้เหมาะ คุณไม่คิดว่าจะจัดการกับลักษณะของหน้าต่าง และไม่คิดว่าจะจัดการกับรายละเอียดของกริด แต่ตั้งใจจะเริ่มโปรแกรม ก็ให้ชื่อโพรซีเดอร์ได้ว่า InitApp ซึ่งย่อและรวมมาจาก Initialize Application นั่นเอง

อันนี้ก็อาจมีประโยชน์มากครับในกรณีที่แอพพลิเคชันของคุณเปลี่ยนรูปทรงไปเรื่อย ๆ ขณะกำลังรัน เกิดรายละเอียดขึ้นมากมายจากการติดต่อของผู้ใช้ แล้วมีสักจังหวะคุณอยากให้มันกลับที่เก่า เข้าสู่จุดเริ่มต้นใหม่ คุณก็แค่สั่ง InitApp ทีเดียว ทุกอย่างก็กลับสู่จุดเริ่มต้นอีกครั้ง เรื่องนี้มีข้อแม้อยู่ว่าคุณต้องเก็บรายละเอียดในการเริ่มโปรแกรมไว้ให้ครบ อย่าให้ขาดตกหกหล่นแม้กระทั่งเรื่องหยุมหยิมที่ก่อผลข้างเคียงได้ใหญ่โต เป็นต้นว่าค่าแฟล็กหรือค่าตัวแปรร่วมทั้งหลายแหล่ พวกมันต้องอยู่ในสภาพพร้อมใช้งานใหม่ในโพรซีเดอร์ InitApp นี้ เทคนิคง่าย ๆ ก็คือเช็คดูให้ละเอียดว่าตัวแปรที่ดีแคลร์ไว้เป็นโกลบอลมีอะไรบ้าง แต่ละตัวทำหน้าที่อย่างไร ต้องเปลี่ยนหรือต้องไม่เปลี่ยนเมื่อเกิดโพรซีเดอร์ InitApp ขึ้นมา

ควรวางแผนในการเขียนโค้ดใน VB อย่างไร?

อันนี้ก็เป็นคำถามที่น่าสนใจมาก อย่างที่เกริ่นไว้ข้างต้นแล้วว่า VB ตอบสนองได้ทั้งโปรแกรมเมอร์มือก๊วยเตี๋ยวเส้นเล็กน้ำ และโปรแกรมเมอร์มือพระกาฬ ไม่ว่าคุณจะชอบทำโค้ดให้ลึกลับด้วยความเมาหน้าจอหรือว่าด้วยแผนสูงลึกล้ำท่าไหนก็แล้วแต่ คุณมีสิทธิ์

เขียนโค้ด VB เป็นภายในเวลาไม่นานนัก

แต่แทนที่จะมะงุมมะงาหราชักช้าไม่ทันกินเขา เอาแต่งมเองอยู่คนเดียว สู้มาดูหลักการการเขียนโปรแกรมใน VB กันดีกว่า อาจช่วยให้ถึงที่หมายได้เร็วและไม่ต้องผจญกับเส้นก๋วยเตี๋ยวของตนเองนานเกินไป

ข้างล่างนี้เป็นเพียงหลักการคร่าว ๆ ที่ไม่ใช่เนื้อหาวิชาการที่ต้องการนิยามมากมายเหมือนในตำราซอฟต์แวร์ดีไซน์ มีไว้ให้อ่านเป็นอนุสติก่อนลงมือเขียนโค้ดด้วย VB ครั้งต่อไปเท่านั้นครับ

1. เขียนใส่กระดาษเป็นคำบรรยายทั่วไปเกี่ยวกับแอพพลิเคชันที่กำลังจะทำว่ามีจุดประสงค์ใด ใช้งานโดยใครเกี่ยวข้องกับออปเจ็กต์หรือ environment ใดบ้าง เขียนเหมือนบอกเล่าให้ตัวเองฟังสบาย ๆ ไม่คิดอะไรเคร่งครัด ปล่อยให้สามัญสำนึกมันพาไปจากจุดเริ่มต้นเอง อันนี้จะช่วยกระตุ้นให้เกิดความรับรู้งานและความคิดออกแบบสร้างสรรค์ ส่วนใหญ่โปรแกรมเมอร์จะเฉื่อยชาต่อการรับรู้งานใหม่ ความคิดสร้างสรรค์ไม่ค่อยเกิด ความเข้าใจและแรงบันดาลใจจึงไม่ค่อยมี เป็นเหตุเป็นผลกันด้วยประการฉะนี้

2. ตัดสินใจเลือกออปเจ็กต์ด้วยความรอบรู้และมองการณ์ไกล มันจะมีผลต่อการออกแบบ การโค้ด รวมทั้งการใช้และการเมนเทนที่จะมีขึ้นอย่างแน่นอนในอนาคตข้างหน้า หากคุณเลือกได้เหมาะสม ปัญหาในทุกขั้นตอนก็จะไม่เกิด หรือเกิดน้อยที่สุด ยกตัวอย่างเช่น เขาต้องการให้สร้างหน้าจอแสดงข้อมูลหลายเรคอร์ดพร้อมกัน ใครผ่าไปเอา Label มานั่งเรียงนั่งประดิดประดอยก็แย่เลยครับ เขามีกริดให้ใช้ดี ๆ ไม่เอา อยากทำงานยาก แถมดูยากอีกต่างหาก เอ๊อ...

3. ตั้งค่า properties อย่างมีศิลปะ ไม่ปล่อยให้เป็นไปตามยถากรรมเกินไปนัก เช่น เขาให้ค่าดีฟอลต์อะไรมาก็รักษาไว้หมด บูชามาก หรือไม่ก็ขี้เกียจตกแต่ง ยกตัวอย่างเช่น ทำแอพพลิเคชันร้อยชิ้นร้อยประเภท พื้นสีฟอร์มก็ขาวโพลนทั้งร้อยชิ้นอยู่นั่นแหละ เป็นสีต้องโฉลกที่หมอดูแนะนำหรือไงก็ไม่ทราบ อันนี้ต้องปรับให้เหมาะกับแอพพลิเคชันกันบ้างครับ การไม่ละเลยเรื่องหยุมหยิมก็อาจเป็นบันไดสู่ความเป็นมืออาชีพได้เหมือนกัน

4. เขียนโค้ดอย่างมีแผน โดยเริ่มสะกัดจาก requirement นั่นแหละ โหลดขึ้นมาปุ๊บเขาต้องการเห็นอะไรก็จัดการเสียให้เสร็จ ไม่ว่าจะเป็นการตั้งค่า properties ตอน design time หรือจะเป็นการเขียนโค้ดในส่วน InitApp ก็เลือกเอาตามอัธยาศัยและความเหมาะด้วยประการทั้งปวง

จากนั้นก็ลิสต์ไว้เป็นข้อ ๆ ว่าเมื่อผู้ใช้กดปุ่มนั้น ลากเม้าส์ไปคลิ๊กทางโน้น หรือเคาะ Enter จะให้เกิดอะไรขึ้นบ้าง เมื่อเกิดแล้วจะมีผลกระทบกับส่วนอื่นที่ไหน ให้ดีก็วงเล็บไว้เสียด้วยว่าแต่ละข้อนั้นมีออปเจ็กต์หรือโพรซีเดอร์ใดเข้ามาเกี่ยวข้องบ้าง

เขียนแล้วเก็บไว้ดี ๆ นะครับ ไม่ใช่ขยำทิ้ง แผ่นกระดาษเล็ก ๆ นี่แหละมีค่านักเมื่อคุณโค้ดไปโค้ดมาแล้วงง หรืออยากได้สิ่งอ้างอิงที่จะนำความคิดให้ทำงานเชื่อมโยงตลอดสาย คุณจะพบว่ามันช่วยให้คุณคิดได้เร็ว วางแผนโปรแกรมได้เป็นระเบียบ รวมทั้งไม่ให้คุณเกิดช่วงเฉื่อยชา อันเนื่องจากอาการได้หน้าลืมหลังซึ่งมักปรากฏเมื่อคุณใส่โค้ดเข้าไปเรื่อย ๆ พอดูอีกทีอ้าวเป็นกองพะเนินเทินทึกเสียแล้ว ต้นอยู่ไหนปลายอยู่ไหนจับไม่ถูก ตาเหล่ลิ้นจุกปาก เอ๊ะทำไมผมรู้อาการพวกนี้ดีจัง?

5. ฝืนใจเขียนคอมเมนต์เยอะ ๆ มันไม่ใช่เรื่องยากหรอกครับกับการอธิบายหมายจำเอาไว้บ้างเมื่อถึงบรรทัดสำคัญที่เกิดการกระทำอันเป็นหัวเลี้ยวหัวต่อทั้งหลาย ก็แค่เอาปลายนิ้วก้อยไปเขี่ยเขี้ยวเสียหน่อย แล้วอธิบายสั้น ๆ ไปว่าบรรทัดนั้นมันทำอะไร เป็นจุดเริ่มต้นวงจรชนิดไหน ขอแค่พออ่านรู้เรื่องของคุณเอง ใครมันจะมาอ่านเจอแล้วงงแปดกลับก็ช่างเขา

ก่อนจบ

เอาล่ะครับ คุยเฟื่องเรื่องเทคนิคเล็ก ๆ น้อย ๆ กันพอหอมปากหอมคอ พบกันใหม่ฉบับหน้าครับ *สวัสดี.*